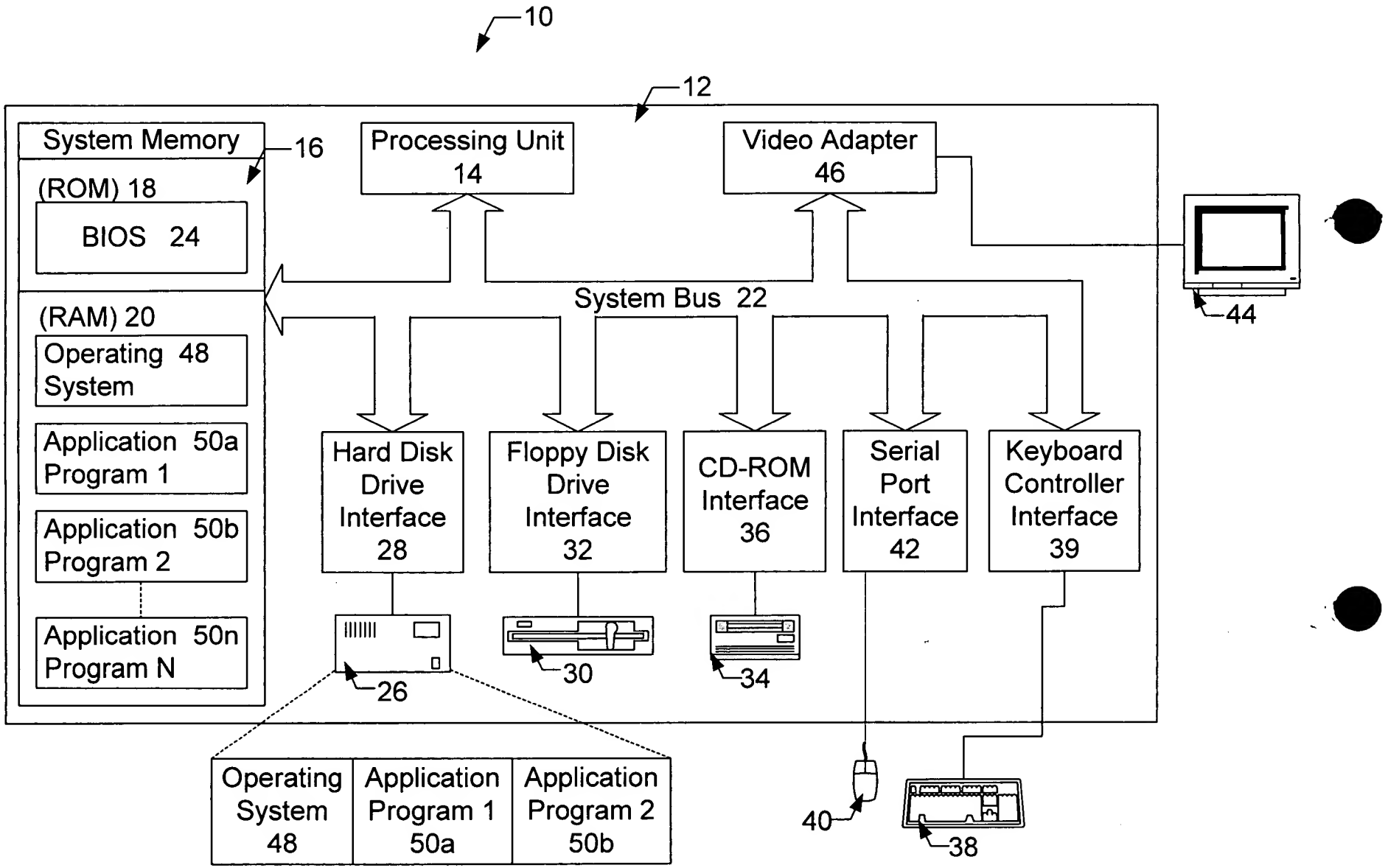
10
12
System Memory
16
(ROM) 18
BIOS 24
(RAM) 20
Operating System 48
Application Program 1 50a
Application Program 2 50b
Application Program N 50n
Processing Unit 14
Video Adapter 46
System Bus 22
44
Hard Disk Drive Interface 28
Floppy Disk Drive Interface 32
CD-ROM Interface 36
Serial Port Interface 42
Keyboard Controller Interface 39
26
30
34
40
38
Operating System 48
Application Program 1 50a
Application Program 2 50b

Figure 1

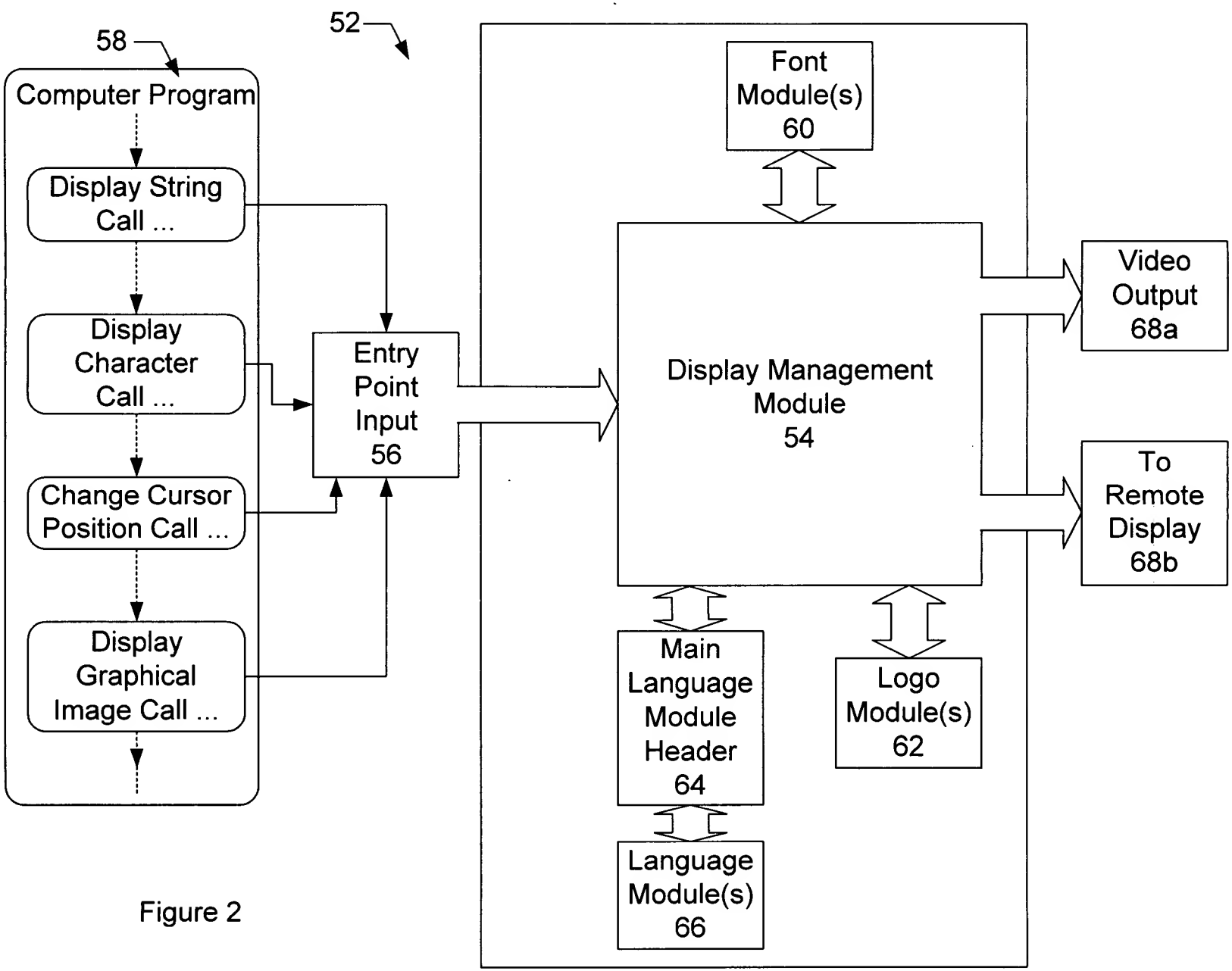

Figure 2

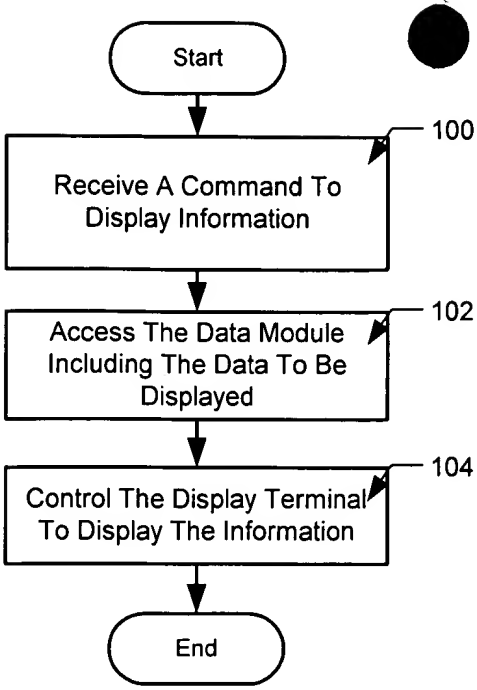

Figure 3

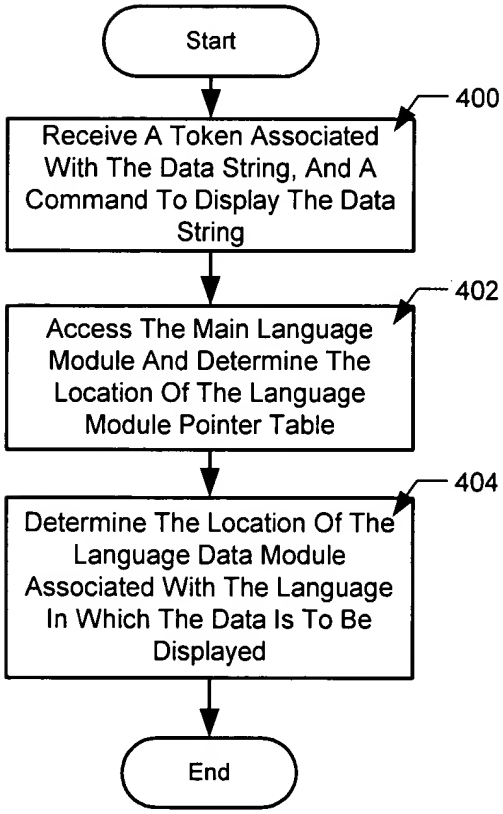

Figure 8

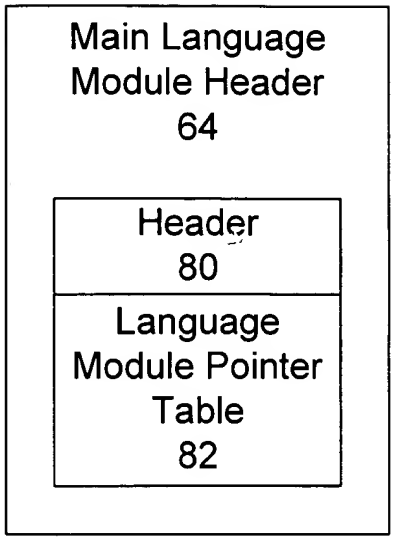

Figure 7

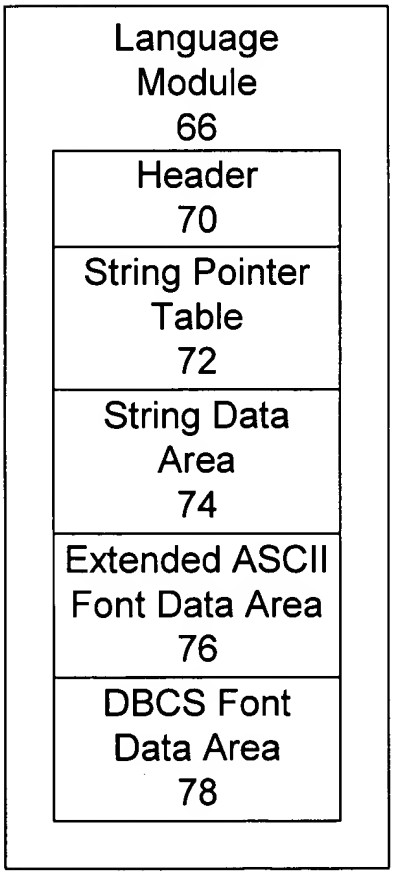

Figure 4

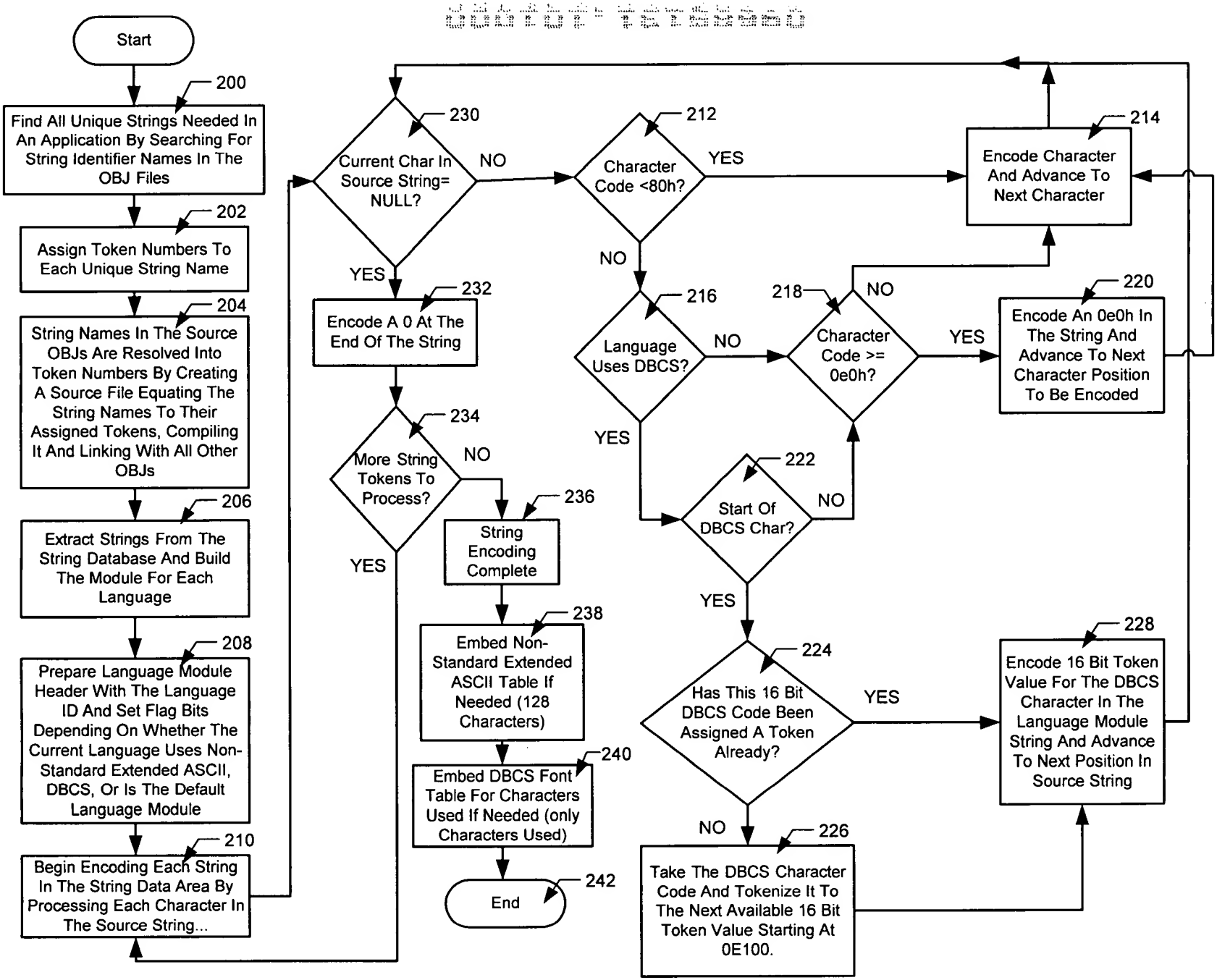
Start
200
Find All Unique Strings Needed In An Application By Searching For String Identifier Names In The OBJ Files
202
Assign Token Numbers To Each Unique String Name
204
String Names In The Source OBJs Are Resolved Into Token Numbers By Creating A Source File Equating The String Names To Their Assigned Tokens, Compiling It And Linking With All Other OBJs
206
Extract Strings From The String Database And Build The Module For Each Language
208
Prepare Language Module Header With The Language ID And Set Flag Bits Depending On Whether The Current Language Uses Non-Standard Extended ASCII, DBCS, Or Is The Default Language Module
210
Begin Encoding Each String In The String Data Area By Processing Each Character In The Source String...
230
Current Char In Source String= NULL?
NO
YES
212
Character Code <80h?
YES
NO
214
Encode Character And Advance To Next Character
216
Language Uses DBCS?
NO
YES
218
Character Code >= 0e0h?
NO
YES
220
Encode An 0e0h In The String And Advance To Next Character Position To Be Encoded
222
Start Of DBCS Char?
NO
YES
224
Has This 16 Bit DBCS Code Been Assigned A Token Already?
YES
NO
226
Take The DBCS Character Code And Tokenize It To The Next Available 16 Bit Token Value Starting At 0E100.
228
Encode 16 Bit Token Value For The DBCS Character In The Language Module String And Advance To Next Position In Source String
232
Encode A 0 At The End Of The String
234
More String Tokens To Process?
NO
YES
236
String Encoding Complete
238
Embed Non-Standard Extended ASCII Table If Needed (128 Characters)
240
Embed DBCS Font Table For Characters Used If Needed (only Characters Used)
242
End

Figure 5

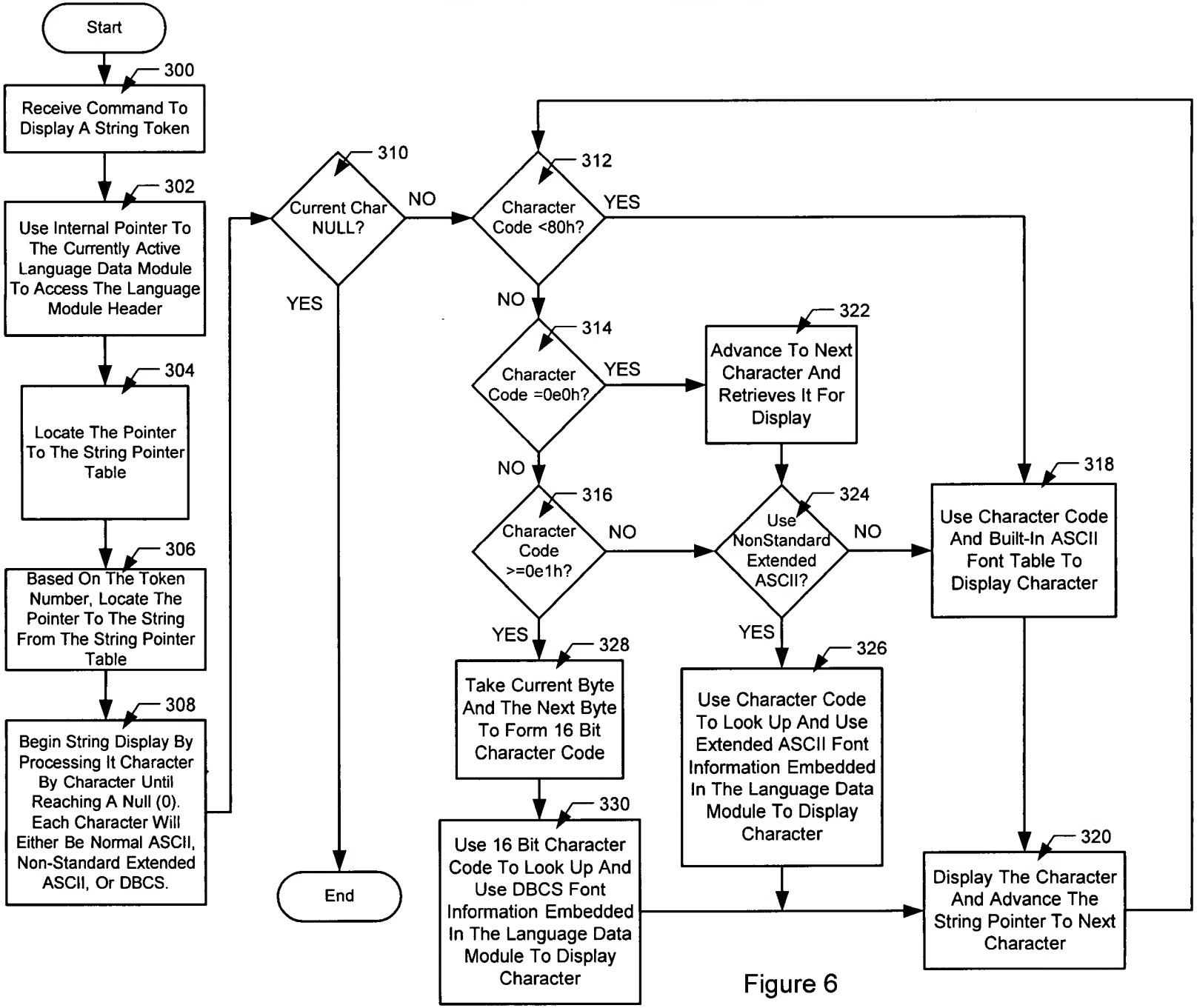
Start
300
Receive Command To Display A String Token
302
Use Internal Pointer To The Currently Active Language Data Module To Access The Language Module Header
304
Locate The Pointer To The String Pointer Table
306
Based On The Token Number, Locate The Pointer To The String From The String Pointer Table
308
Begin String Display By Processing It Character By Character Until Reaching A Null (0). Each Character Will Either Be Normal ASCII, Non-Standard Extended ASCII, Or DBCS.
310
Current Char NULL?
NO
YES
End
312
Character Code <80h?
YES
NO
314
Character Code =0e0h?
YES
NO
316
Character Code >=0e1h?
NO
YES
322
Advance To Next Character And Retrieves It For Display
324
Use NonStandard Extended ASCII?
NO
YES
318
Use Character Code And Built-In ASCII Font Table To Display Character
326
Use Character Code To Look Up And Use Extended ASCII Font Information Embedded In The Language Data Module To Display Character
328
Take Current Byte And The Next Byte To Form 16 Bit Character Code
330
Use 16 Bit Character Code To Look Up And Use DBCS Font Information Embedded In The Language Data Module To Display Character
320
Display The Character And Advance The String Pointer To Next Character

Figure 6

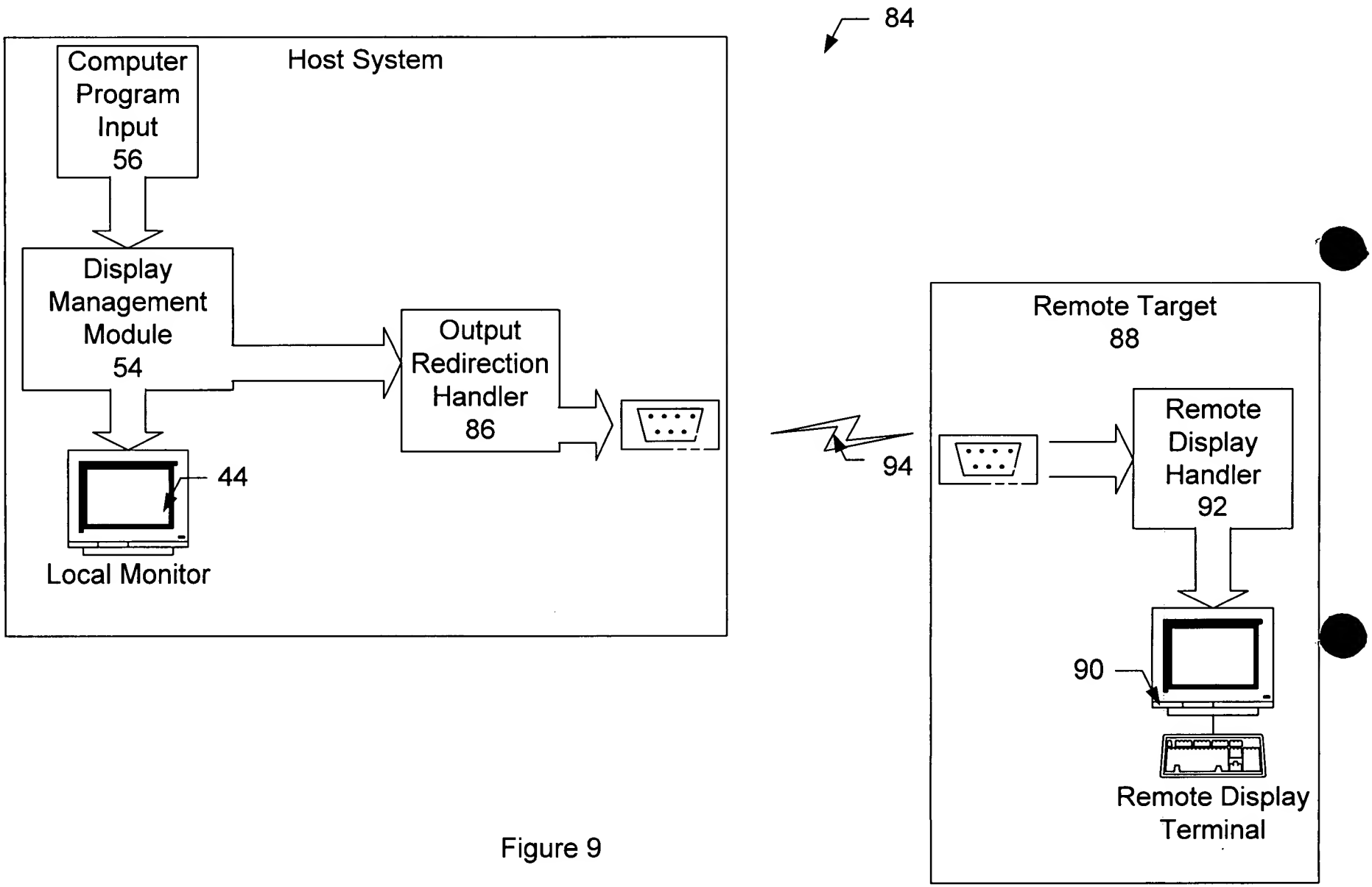

Figure 9

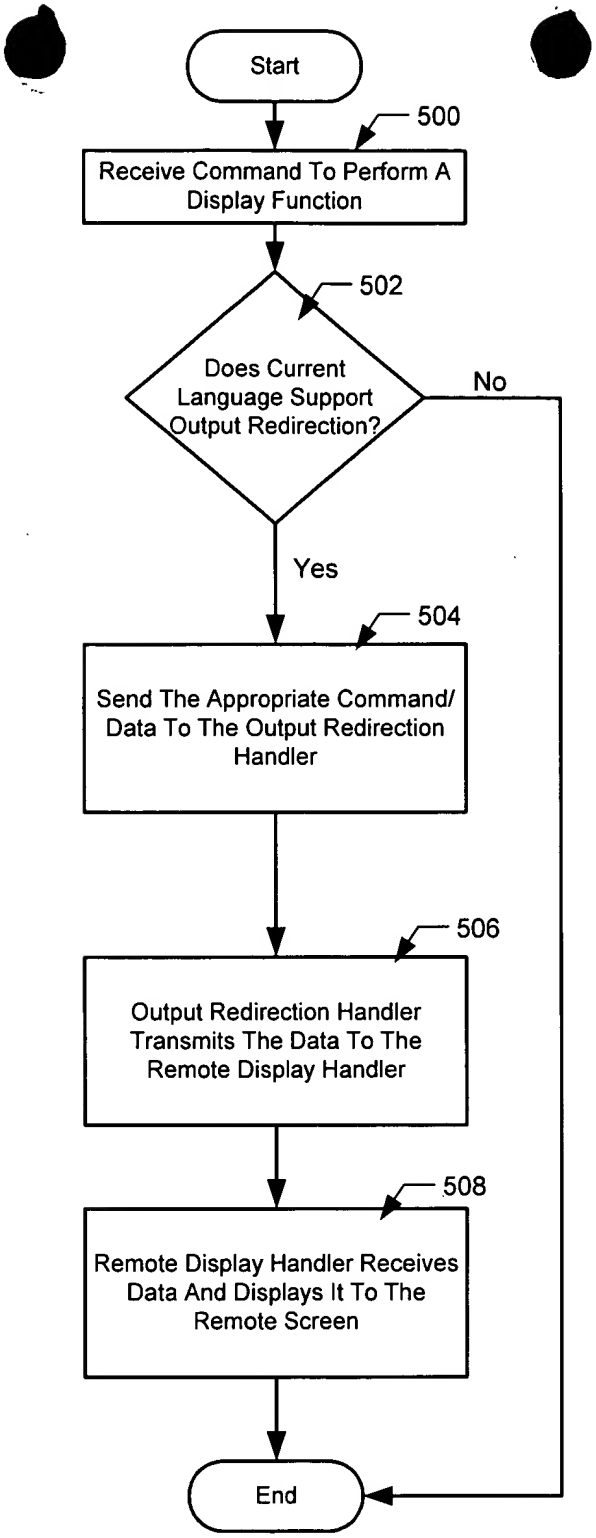
Start
500
Receive Command To Perform A Display Function
502
Does Current Language Support Output Redirection?
No
Yes
504
Send The Appropriate Command/ Data To The Output Redirection Handler
506
Output Redirection Handler Transmits The Data To The Remote Display Handler
508
Remote Display Handler Receives Data And Displays It To The Remote Screen
End

Figure 10